เทคนิคเกี่ยวกับดีเบสทรีพลัส

ยืน ภู่วรวรรณ

ในฉบับนี้ผู้เขียนขอรวบรวมเทคนิคต่าง ๆ ที่เป็นประโยชน์มาไว้รวมกันในที่นี้ เทคนิคเหล่านี้บางส่วนเกิดจากประสบการณ์ที่ผู้เขียนพบ บางส่วนเกิดจากการค้นหาจากหนังสือของต่างประเทศ ผู้เขียนคิดว่าน่าจะเป็นประโยชน์โดยรวม จึงได้นำมารวบรวมเผยแพร่ให้ผู้สนใจได้มีโอกาสนำเอาไปใช้งานได้

ทำอย่างไรจึงจะใช้งานดีเบสทรีพลัสได้ดีในระบบไมโครคอมพิวเตอร์ที่มีหน่วยความจำเพียง 256 K

สืบเนื่องจากดีเบสทรีพลัสมีการกันหน่วยความจำไว้ใช้งานอย่างอื่นอยู่มาก จึงทำให้ผู้ใช้ใช้งานได้จำกัดในเครื่องคอมพิวเตอร์ที่มีหน่วยความจำ 256 K เช่น เปิดไฟล์ได้เพียง 2 ไฟล์ พอเปิดไฟล์ที่ 3 ก็จะไม่ยอมให้เปิด โดยขึ้นข้อความ too many file are opened

อย่างไรก็ดีเราสามารถใช้งานในระบบ 256 K ได้สมบูรณ์พอควร โดยเปิดไฟล์ได้ครบตามข้อกำหนดของดีเบสทรีพลัสได้ และสามารถทำงานประยุกต์ได้สมบูรณ์ โดยผู้ใช้จะต้องกระทำตามเงื่อนไขดังนี้

1. เอ็มเอสดอสที่ใช้จะใช้ได้เฉพาะ MS-DOS เวอร์ชัน 2 หรือ 2.1 เท่านั้น จะใช้เวอร์ชัน 3 หรือมากกว่าไม่ได้ เพราะ MS-DOS เวอร์ชัน 3 จะคงส่วนของ COMMAND.COM ไว้ในหน่วยความจำ ทำให้เสียพื้นที่ไปทำให้เนื้อที่ที่เหลือไม่พอใช้สำหรับดีเบสทรีพลัส

2. จะต้องใช้ CONFIG.SYS ในการบูตระบบด้วยเงื่อนไขดังนี้

FILES = 20
BUFFERS = 4

เป็นการลดขนาดบัฟเฟอร์ลงจาก 24 เหลือ 4 ผลจะทำให้การอ่านดิสก์ที่มีข้อมูลจำนวนมากช้าลง เช่น คำสั่ง LIST จะช้าลงบ้าง

3. จะต้องลดขนาดของพื้นที่ใช้งานบางส่วนของดีเบสทรีพลัสลงบ้าง โดยทำการติดตั้ง CONFIG.DB ไว้ในแผ่น System 1 โดยมีข้อความดังนี้

```
COMMAND   = ASSIST
STATUS    = ON
BUCKET    =  1
GETS      = 35
MVARSIZ   =  3
HISTORY   = 10
HISTORY   = 10
TYPEAHEAD = 10
```

ความหมายของการติดตั้งระบบในส่วนนี้มีความหมายดังนี้

COMMAND = ASSIST หมายถึงให้เข้า ASSIST เมื่อเริ่มต้น หากไม่ต้องการจะตัดออกก็ได้ แต่เมื่ออยู่ในโหมด ASSIST ก็สามารถใช้ ESC คีย์ เลิกมาที่ . ได้

STATUS = ON หมายถึงให้มีบรรทัดสถานะที่อยู่ตอนล่าง

BUCKET = 1 คือเตรียมพื้นที่สำหรับ PICTURE และ RANGE เท่ากับ 1×1024 ไบต์ ปกติดีเบสทรีพลัสเตรียมไว้ 2

GETS = 35 คือเตรียมพื้นที่สำหรับรับคำสั่ง GET 35 คำสั่ง โดยปกติดีเบสทรีพลัสเตรียมไว้ 128

MVARSIZ = 3 คือเตรียมพื้นที่สำหรับตัวแปรจำนวน 3 K โดยปกติดีเบสทรีพลัสเตรียมไว้ 6 K

HISTORY = 10 คือเตรียมเก็บประวัติคำสั่งไว้ 10 คำสั่ง หลังจากที่คีย์ใน . โดยปกติดีเบสทรีพลัสเตรียมไว้ 20

TYPEAHEAD = 10 คือรับคีย์บอร์ดได้ล่วงหน้า 10 ตัวอักษร ดีเบสทรีพลัสเตรียมไว้ในภาวะปกติ 16 ตัว

ข้อจำกัดอีกประการหนึ่งที่จะเกิดกับระบบที่ใช้หน่วยความจำ 256 K คือ จะใช้คำสั่ง RUN ไม่ได้ เพราะคำสั่งจะต้องโหลด COMMAND.COM ลงมา ซึ่งหน่วยความจำไม่พอเพียงแล้ว

ข้อแนะนำของดีเบสคือ ควรใช้หน่วยความจำไม่น้อยกว่า 384 K แต่อย่างไรก็ตามถ้าใช้ตามข้อแนะนำข้างบนนี้ก็จะใช้งานได้ตามปกติ

ดีเบสทรีพลัสกับคำสั่ง EXIT และ LOOPในลูป DO WHILE

เมื่ออยู่ในลูป DO WHILE จะมีการตรวจสอบเงื่อนไขที่ do while ถ้าเป็นจริงก็จะกระทำตาม แล้ววนกลับมาตรวจสอบที่ DO WHILE ใหม่ (เมื่อพบ ENDDO) หากเงื่อนไขเป็นเท็จก็จะออกจากลูปไปคำสั่งต่อไป

ดีเบสทรีพลัส ยอมให้ผู้โปรแกรมมาควบคุมการกระโดดจากคำสั่งในลูปมายัง DO WHILE โดยไม่ต้องผ่านคำสั่ง ENDDO ได้ โดยใช้คำสั่ง LOOP ลองดูจากตัวอย่างในรูปที่ 1

คำสั่ง LOOP ในที่นี้จะเป็นการทำให้การทำงานย้อนกลับไปที่ DO WHILE ใหม่อีกโดยไม่ต้องกระทำคำสั่งถัดไป

แต่ถ้าต้องการให้ออกจากลูปในขณะอยู่กลางลูป เราจะใช้คำสั่ง EXIT ซึ่งจะกระโดดมาคำสั่งถัดจาก ENDDO ทันทีดังโปรแกรมในรูปที่ 2

คำสั่ง EXIT ไม่ทำให้เลิกโปรแกรม แต่จะหลุดออกจากลูป DO WHILE ทำให้สามารถใช้งานง่ายขึ้นในบางโอกาส

```
DO WHILE .T.
  DO CASE
    CASE selection = "1"
      DO   (ชื่อโปรแกรม)
    CASE selection = "2"
      *if your balance is 0
      IF Blance = 0
        ? "You're broke"
      ELSE
        * get out of the loop
        EXIT
      ENDIF
  OTHERWISE
      * (command)
  ENDCASE
ENDDO
```

รูปที่ 2 การใช้คำสั่ง EXIT ออกจากลูป

รู้จักใช้ WAIT

ผู้ใช้ดีเบสคงเคยใช้ WAIT กันมาแล้ว เมื่อใช้คำสั่ง WAIT ดีเบสจะหยุดและขึ้นข้อความบนจอเป็นดังนี้

Press any key to continue...

โดยรอให้ผู้ใช้กดคีย์ใดคีย์หนึ่งเพื่อทำงานต่อ การแสดงผลจะขึ้น ณ บรรทัดต่อไปจากตำแหน่งเคอร์เซอร์ โดยเริ่มที่คอลัมน์แรกเสมอ

หากต้องการให้มีข้อความบางอย่างขึ้นแทน เราใช้คำสั่ง WAIT ได้ดังนี้

WAIT SPACE (19) + "Press any key to retum to the Main Menu"

ในกรณีที่จะทำให้ข้อความมาอยู่กลางบรรทัดโดยเว้นไป 19 คอลัมน์ก่อนแล้วพิมพ์ข้อความออกมาบนจอภาพ

นอกจากนี้เราอาจจะมีการรับคีย์เข้ามายังตัวแปรเพื่อตรวจสอบได้ ลองดูจากตัวอย่างในรูปที่ 3

การใช้ RANGE ในการรับข้อมูล

ในการใช้คำสั่ง GET รับข้อมูล เราสามารถรับข้อมูลตัวเลขที่มีช่วงที่กำหนดได้ โดยใช้คำสั่ง Range เช่น ต้องการรับข้อมูลที่อยู่ในช่วง 25.00 ถึง 100.00 โดยรับเข้ามายังตัวแปรชื่อ amount เราใช้คำสั่ง

@ 10, 10 get amount range 25, 100

แต่ถ้าต้องการกำหนดว่าข้อมูลต้องไม่ต่ำกว่า 25 เราใช้

@ 10, 10 get amount range 25,

```
* Loop as long as the record pointer is not at the
* end of the file
DO WHILE .NOT. EOF( )
    *IF the record is already marked fer deletion
    IF DELETED ( )
        *move to the next record and start again
        SKIP
        LOOP
    ELSE
        * (many more command)
    ENDIF
ENDDO
```

รูปที่ 1 ตัวอย่างการกระโดดออกจากลูปมายัง DO WHILE

```
WAIT "Press R to return to the main menu,
any other key " + ;  "to continue" TO mchoice
IF UPPER (mchoice) = 'R'
  RETURN
ELSE
  * (command)
ENDIF
```

รูปที่ 3 ตัวอย่างการใช้ WAIT รับคีย์เพื่อตรวจสอบ

หรือกำหนดว่าข้อมูลต้องไม่มากกว่า 100 ก็ใช้

```
@ 10, 10 get amount range, 100
```

การป้อนข้อมูลนี้ยังใช้กับฟอร์แมตวันที่ได้เช่นกัน ลองดูตัวอย่าง

```
@ 10, 10 get m_date Range CTOD ("04/01/87"), CTOD ("04/30/87")
```

การสร้าง PASSWORD

ในการสร้าง PASSWORD และบันทึกผู้ใช้งาน โดยอาจจำกัดจำนวนผู้ใช้ให้อยู่ในกลุ่มที่รู้จักรหัสผ่าน และบันทึกชื่อผู้ใช้และวันเดือนปี และเวลาของผู้ใช้ไว้ด้วย ลองดูจากตัวอย่างต่อไปนี้

สร้างแฟ้มข้อมูลชื่อ HIST.DBF โดยมีโครงสร้าง

USERNAME	C	20
DATE	D	8
CTIME	C	5

เมื่อสร้างแฟ้มเสร็จแล้ว เราจะเก็บโครงสร้างนี้ไว้ ถ้ามีผู้ใช้ผ่านรหัสผ่านได้ เราจะบันทึกชื่อ วันที่และเวลาของผู้ใช้ลงแฟ้ม เพื่อบันทึกการใช้งานโปรแกรมนี้ไว้ ลองดูการกำหนดรหัส password

เราเริ่มจาก . ของดีเบส ลองสร้างตัวแปร

```
.secret = 'ALIBABA'
```

ในที่นี้เราจะตั้งรหัสผ่านของเราว่า ALIBABA เราจะเก็บเป็นแฟ้มชื่อ SECRET.MEM โดยใช้

```
.SAVE TO SECRET
```

ในที่นี้ถ้าขอดูไดเรคทอรี่จะมี SECRET.MEM ถ้าหากไม่ต้องการให้เห็น เราจะใช้ NORTON Utility ตั้งเป็น hidden file ก็ได้

ในขณะป้อน password เราไม่ควรให้ปรากฏบนจอด้วย ในที่นี้จึงใช้ฟังก์ชัน INKEY ( ) มาช่วย เพราะถ้าใช้ ACCEPT หรือ GET จะมีข้อความที่คีย์ปรากฏบนจอภาพด้วย ลองดูโปรแกรม Procedure PASSWORD ในรูปที่ 4

ในที่นี้ เราจะรับคีย์ทีละตัวอักษรมาประกอบกันโดยผ่านทาง inkey ( ) ซึ่งจะไม่ปรากฏบนจอภาพ และมาเปรียบเทียบกับตัวแปร secret ที่เก็บไว้ใน SECRET.MEM

หากจะไม่ใช้ inkey ( ) เรามีทางออกอีกทางหนึ่ง โดยใช้คำสั่ง set color to, N/N หมายถึงไม่แสดงผลหรือเป็นตัวอักษรสีเดียวกับสีแบคกราวนด์โดยใช้คำสั่งในรูปที่ 5

การคำนวณขนาดของไฟล์และพื้นที่ที่เหลือบนดิสค์

ในการทำงานเราควรจะต้องรู้ขนาดของไฟล์ที่ใช้งาน และหากใช้ดีเบสทรีพลัสทำการ SORT หรือ INDEX จำเป็นจะต้องมีพื้นที่บนดิสค์พอเพียง เช่น เมื่อใช้แฟ้ม .DBF ที่มีขนาดใหญ่ เช่น ขนาด

```
Procedure PASSWORD
Clear
USE HIST
MNAME = space (20)
@ 10, 10 say "Enter your name:"
get mname
@ 12, 10 say "Enter your password;"
READ
@ 12, 30
i = 0
password = ""
do while i < 7
  j = 0
  do while j = 0
    j = inkey ( )
  enddo
  password = password + chr (j)
  i = i+1
  if j = 13
    i = 7
  endif
enddo
restore from secret additive
if upper (password) <> secret
  quit
endif
append blank
replace username with mname
replace date with date ( )
replace ctime with substr (now, 1, 5)
use
return
```

รูปที่ 4 โปรแกรม PASSWORD

```
clear
store space (7) to password
* set enhanced display to black on black
set color to, N/N
@ 12, 10 SAY "enter you password:" get password
READ
* revert to normal color
SET COLOR TO
* (command)
```

รูปที่ 5
การใช้ SET COLOR TO แทน INKEY ( )

ประมาณ 1 MB เมื่อทำการ SORT หรือ INDEX จะต้องใช้เวลามาก แต่เมื่อทำไปเป็นเวลานานแล้ว ปรากฏข้อความบนจอว่า พื้นที่ดิสค์ไม่พอ เราจะต้องเสียเวลามากในการรอคอยและจัดการใหม่ ซึ่งน่าจะตรวจสอบได้ในขั้นต้น หรือผู้เขียนโปรแกรมจะตรวจสอบเสียก่อน

ดีเบสทรีพลัสจึงสร้างฟังก์ชันสำหรับจัดการคำนวณหาพื้นที่เหล่านี้ โดยมีฟังก์ชันพิเศษ ดังนี้

RECCOUNT ( ) เป็นฟังก์ชันที่จะให้ค่าจำนวนเรคอร์ดของไฟล์ที่กำลังใช้

RECSIZE ( ) เป็นฟังก์ชันที่จะบอกขนาดของเรคอร์ดเป็นจำนวนไบต์

DISKSPACE ( ) เป็นฟังก์ชันที่บอกขนาดของพื้นที่ ๆ เหลือ

ตัวอย่างของการใช้ RECCOUNT ( ) ในรูปที่ 6 เพื่อจะบอกจำนวนเรคอร์ดเพื่อเตือนให้ทำงานบางอย่าง การนับจำนวนเรคอร์ดนี้จะนับรวมเรคอร์ดที่มี * หรือที่ได้ delete แล้วไว้ด้วยถึงแม้ว่าจะได้ทำการ SET DELETE ON ไว้แล้วก็ตาม

นอกจากนี้ เรายังสามารถคำนวณหาขนาดของแฟ้มข้อมูล .dbf ได้ด้วยการหาขนาดของเรคอร์ดคูณกับจำนวนเรคอร์ด โดยใช้

```
Use persons
size = RECCOUNT ( ) * RECSIZE ( )
```

ขนาดนี้ มิใช่ขนาดของแฟ้ม .dbf จริง ๆ เพราะแฟ้ม .dbf จะมีส่วนของ header อยู่ด้วย เราสามารถคำนวณหา header ได้ โดยต้องทราบว่าแฟ้ม .dbf นี้มีกี่ฟิลด์ โดยใช้การตรวจสอบด้วยฟังก์ชัน FIELD ( ) ดังรูปที่ 7

ขนาดของ header จะเท่ากับจำนวนฟิลด์ * 32+34 ดังนั้น

header = (32*numfield) + 34

```
Use Persons
IF reccount ( ) > 1000
    ? CHR (7)
    @ 10, 10 say 'Time to backup this file'
    * (       )
ENDIF
```

รูปที่ 6 การใช้ RECCOUNT ( ) บอกจำนวนเรคอร์ด

```
Use persons
numfield = 0
null = ""
do while null < field (numfield + 1)
    numfield = numfield + 1
enddo
use
```

รูปที่ 7 การใช้ฟังก์ชัน FIELD ( ) คำนวณหา header

```
use file
* determine size of file and store in variable
* totalsize
* see aboue example cesing field ( ), reccount ( )
* and recsize ( )
*
IF DISKSPACE ( ) < totalsize * 2
    ?-chr (7)
    @ 10, 10 SAY "there isn't enought room on; the disk to sort
          this file"
ELSE
    SORT ON name to temp
ENDIF
```

รูปที่ 8 โปรแกรมตรวจสอบพื้นที่ก่อนทำการ SORT

เมื่อเป็นเช่นนี้ เราคำนวณขนาดของแฟ้ม .dbf ได้อย่างถูกต้องได้

totalsize = size + header + 1

เรานำเอาจำนวนขนาดของแฟ้มและที่เหลือมาใช้ประโยชน์ได้ เช่น ถ้าต้องการจะ SORT เราจะตรวจสอบว่ามีพื้นที่เหลือพอเพียงการ SORT หรือไม่ ดังรูปที่ 8

ในการเพิ่มฟังก์ชันเหล่านี้ มาในดีเบสทรีพลัส ทำให้การใช้งานเพิ่มขึ้นและเป็นประโยชน์ยิ่งขึ้น